User's Manual

Accuracy Meets Speed

Microsoft, Windows, Windows NT, Windows 2000, Windows XP, Visual Basic, Microsoft .NET, Visual C++, Visual C#, and ActiveX are either trademarks or registered trademarks of Microsoft Corporation.

All other nationally and internationally recognized trademarks and tradenames are hereby recognized.

Copyright © 2000-2008 by MVTec Software GmbH, München, Germany

Edition 1	November 2000	(ActivVisionTools 1.0)
Edition 2	April 2001	(ActivVisionTools 1.3)
Edition 3	September 2001	(ActivVisionTools 2.0)
Edition 4	November 2002	(ActivVisionTools 2.1)
Edition 5	January 2005	(ActivVisionTools 3.0)
Edition 6	February 2006	(ActivVisionTools 3.1)
Edition 7	May 2008	(ActivVisionTools 3.2)

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

More information about ActivVisionTools can be found at:

`http://www.activ-vision-tools.com`

How to Read This Manual

This manual explains how to use ActivMeasure to measure distances between edges along lines or arcs. It describes the functionality of ActivMeasure and its cooperation with other ActivVisionTools with Visual Basic examples. Before reading this manual, we recommend to read the manual Getting Started with ActivVisionTools, which introduces the basic concepts of ActivVisionTools and the User's Manual for ActivView to learn how to load and display images.

To follow the examples actively, first install and configure ActivVisionTools as described in the manual Getting Started with ActivVisionTools. For each example in this manual, there is a corresponding Visual Basic project; these projects can be found in the subdirectory `examples\manuals\activmeasure` of the ActivVisionTools base directory you selected during the installation (default: `C:\Program Files\MVTec\ActivVisionTools`). Of course, you can also create your own Visual Basic projects from scratch.

We recommend to **create a private copy of the example projects** because by experimenting with the projects, you also change their *state*, which is then automatically stored in the so-called description files (extension `.dsc`) by ActivVisionTools. Of course, you can restore the state of a project by retrieving the corresponding description file from the CD.

Contents

1 About ActivMeasure **1**

1.1 Introducing ActivMeasure 2

1.2 The Sub-Tools of ActivMeasure 4

2 Using ActivMeasure **7**

2.1 Specifying the Line of Measurement 8

2.2 Extracting Edges 10

2.3 Selecting Results 14

3 Combining ActivMeasure with other ActivVisionTools **17**

3.1 Converting Results to Other Units 18

3.2 Evaluating Results 20

3.3 Output of Results 22

4 Tips & Tricks **25**

4.1 Adapting the Display of Results 26

4.2 Configuring the Two Execution Modes 28

4.3 Accessing Results Via the Programming Interface 30

Chapter 1

About ActivMeasure

This chapter will introduce you to the features and the basic concepts of ActivMeasure. It gives an overview about ActivMeasure's *master tool* and its *support tools*, which are described in more detail in chapter 2 on page 7.

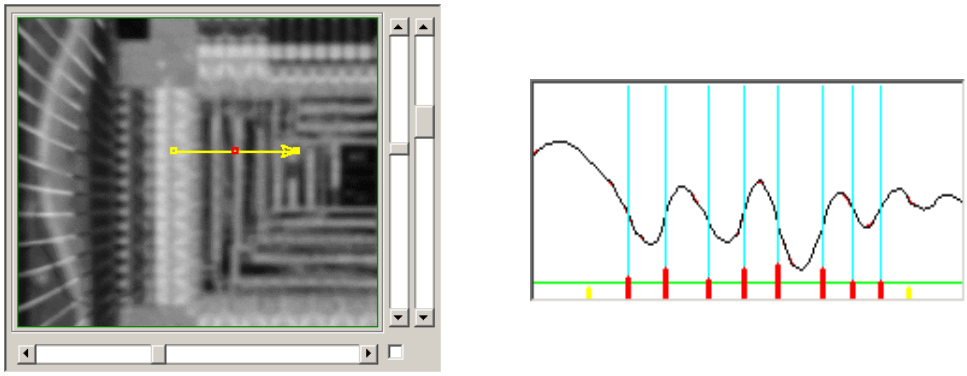
1.1 Introducing ActivMeasure 2
1.2 The Sub-Tools of ActivMeasure 4

1.1 Introducing ActivMeasure

With ActivMeasure, you can measure the distance between *edges* along a line or an arc. Such measurements are a typical task in automated quality control (e.g., chip inspection). ActivMeasure assures subpixel accuracy at high speed, even when taking measurements in multiple ROIs.

Edges

Edges are strong discontinuities in pixel values. This is illustrated in figure 1.1: On the left you see a (zoomed) image of a printed circuit board, with a line-shaped ROI for measuring. On the right, the so-called *line profile* is displayed, i.e., the pixel values along that line. Note that dark pixels correspond to low values. In such a profile, edges correspond to points of a strong change of pixel values. They are marked by vertical lines; the bold parts at the bottom correspond to their strength (amplitude).



a) b)

Figure 1.1: (a) Image with line of measurement; (b) corresponding gray value line profile.

Measurement Paths

In the example above, edges were searched for along a line. Besides this, ActivMeasure also allows measurements along a circular arc. To extract edges more robustly, it is useful to "widen" the examined image region, e.g., from a line to a rectangle and to average pixel values along that dimension, i.e., perpendicularly to the line of measurement. However, one has to take care then that the edges one want to measure are almost parallel to the direction of averaging.

In figure 1.2 for example, the goal is to measure the non-parallel wires. If the pixel values are

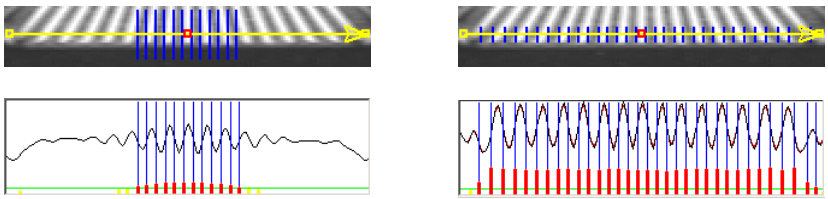


Figure 1.2: Edge setection with and without averaging.

averaged (over a width of 10 pixels), the outer edges are are “smoothed down”, as can be seen in the corresponding profile. Averaging can also be performed for measurements along arcs.

If you want to measure an object which moves from image to image, the line or arc of measurement must be moved accordingly. For this, you need the tool ActivAlignment. Please consult the corresponding user’s manual for more information.

Modes of Measuring

Besides measuring the distances between successive edges, you can choose to group edges to pairs and measure the distances between them, or measure the distance between the first and the last edge. Additional results (also called *features*) that can be calculated are the center position and the width of a pair and the distance between pair centers (see figure 1.3).

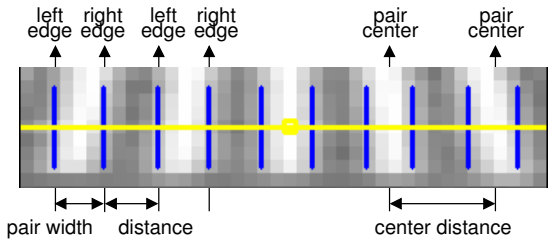


Figure 1.3: What can be measured between edge pairs.

If you calibrated the size of a pixel using either ActivView’s support tool `AVTViewCalibration` or ActivGeoCalib, measured distances and positions can be converted to other units besides pixels.

It is also possible to measure angles of edges and angular distances between successive edges. If the edges are grouped to pairs, additionally, the angular pair width, the angle of the pair centers, and the angular distance between successive pair centers can be calculated (see figure 1.4).

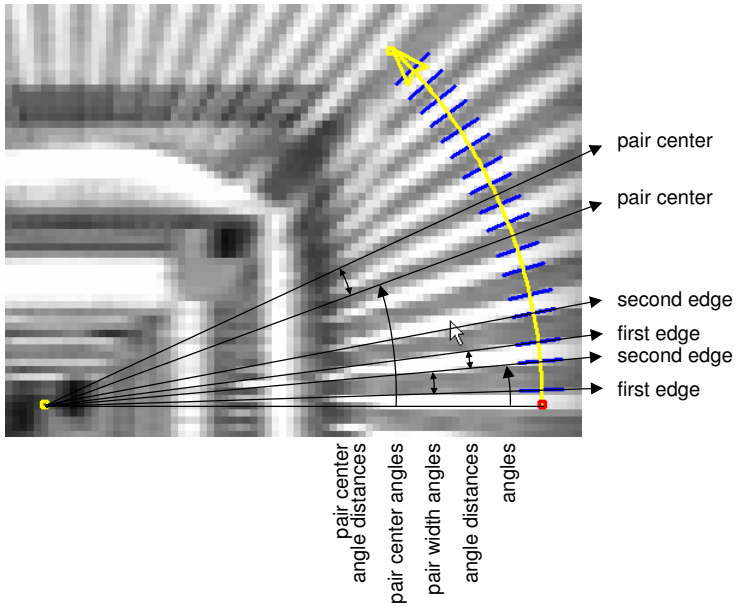


Figure 1.4: Which angular features can be calculated.

Note that the angular features provide meaningful information only for arcs of measurement because the direction of the edges is always perpendicular to the line or arc of measurement. For this reason, all angular features of measurement lines are equal to zero. The angular features of measurement arcs are derived from the angles of the displayed edges, which are perpendicular to the measurement arc. These edges are not necessarily identical to the edges in the image.

Output of Results

Measured features can be displayed using ActivDataView or ActivFeatureHistogram and then further evaluated using ActivDecision, before you output them via ActivFile, ActivSerial, or ActivDigitalIO; furthermore, you can access results via the programming interface (see section 4.3 on page 30).

1.2 The Sub-Tools of ActivMeasure

Besides its *master tool*, ActivMeasure provides 2 *support tools*. In figure 1.5 they are depicted together with other ActivVisionTools that you will use in a typical ActivMeasure application.

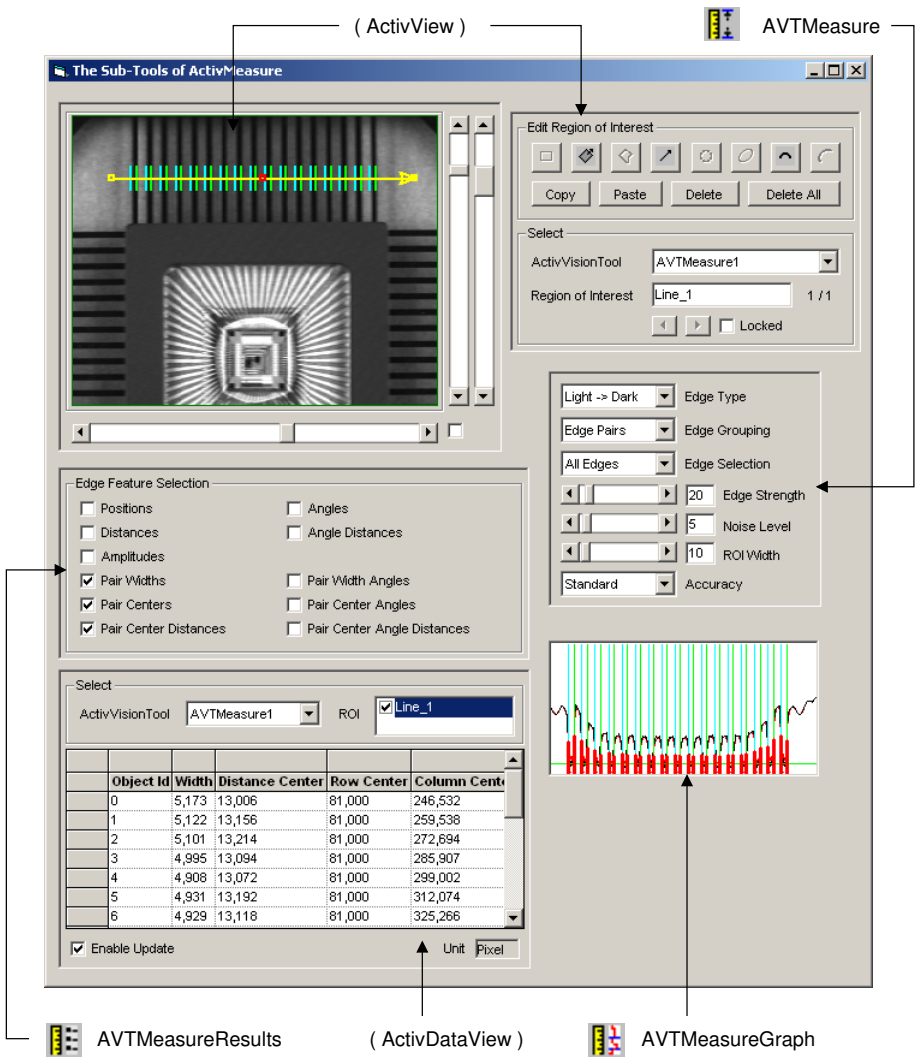


Figure 1.5: The sub-tools of ActivMeasure together with suitable other tools.

`AVTMeasure` is the *master tool* of ActivMeasure. With it, you can select which kind of *edges* are to be extracted and whether they are to be grouped to pairs. Furthermore, you can control image processing to make sure that you get those edges you want.

How to use `AVTMeasure` is described in more detail in section 2.2 on page 10.

`AVTMeasureGraph` is a *support tool* of ActivMeasure. It displays the *line profile*, i.e., the pixel values along the line of measurement together with the position of the edges. Thereby, it helps you to select suitable image processing parameters.

How to use `AVTMeasureGraph` is described in section 2.2 on page 10.

`AVTMeasureResults` is a *support tool* of ActivMeasure. It allows you to select which measurements are to be performed.

How to use `AVTMeasureResults` is described in section 2.3 on page 14.

Chapter 2

Using ActivMeasure

This chapter will explain how to use ActivMeasure to measure distances between edges along lines or arcs. It describes how to create the different types of "lines of measurement", how to parameterize edge extraction, and the types of measurements that can be taken.

The corresponding Visual Basic projects show how to solve an example measuring task.

2.1 Specifying the Line of Measurement 8
2.2 Extracting Edges 10
2.3 Selecting Results 14

2.1 Specifying the Line of Measurement Using

ActivMeasure lets you choose between two shapes of measurement “lines”: straight lines or circular arcs. To get more robust results, you can let ActivMeasure average pixel values in a certain margin perpendicular to the line or arc. To facilitate the creation of very wide averaging margins along a line, you can directly create a rectangle. You specify a line of measurement using `AVTViewROI`, which is a *support tool* of ActivView.

Visual Basic Example

Preparation for the following example:

- ☐ Open the project `rois\measure_rois.vbp`. Alternatively, create a new project and place the following tools on the form (in parentheses the icon you have to double-click with the left mouse button): `AVTView` ( ), `AVTViewROI` ( ), and `AVTMeasure` ( ).
- ☐ Execute the application (`Run` ▷ `Start` or via the corresponding button).
- ☐ Open `AVTViewFG` by clicking on `AVTView` with the right mouse button and selecting `Image Acquisition`. Load the image `chip\chip_01` via the combo box `Input File`.

The following steps are visualized in figure 2.1.

① First, you have to tell `AVTViewROI` that it is to create an ROI for ActivMeasure by selecting the corresponding entry in the combo box `ActivVisionTool`. In this box, all ROI-processing ActivVisionTools are listed that have been placed upon the form. The tools are referenced by the *name* of the corresponding ActiveX control. In the default behavior of Visual Basic, multiple instances of a control are distinguished by a postfix counter, e.g., `AVTMeasure1` and `AVTMeasure2`. In our example, there is only one item in the combo box, `AVTMeasure1`.

② Next, select the desired shape of the line of measurement.

③ To draw the line of measurement, move the mouse in the image while keeping the left mouse button pressed. Please experiment at this point with the different shapes.

④ Now, you can move the ROI by dragging its pick point in the middle. By dragging the outer pick points you modify its shape. Again, please experiment to get familiar with the ROIs.
To finely position an ROI, you can zoom the image and move the displayed part using the scrollbars to the right and below the image.

⑤ To create a second ROI, repeat step ②. You can also copy, paste, and delete the selected

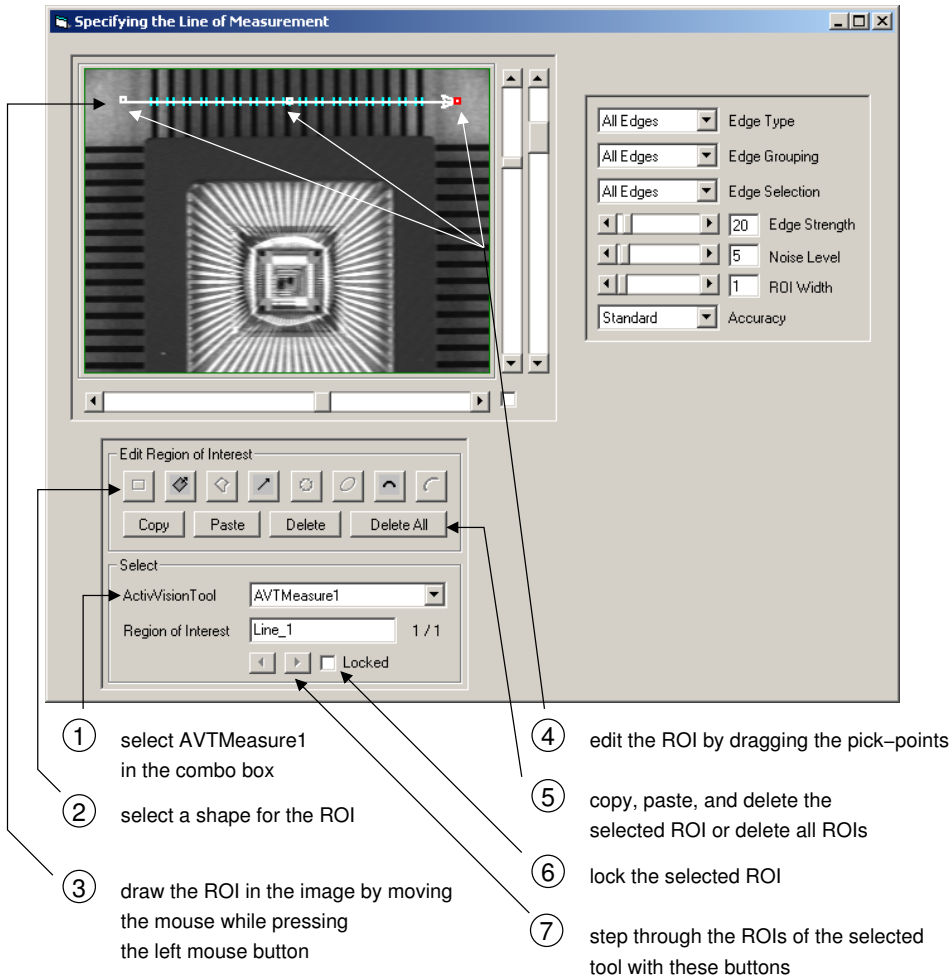


Figure 2.1: Creating the lines of measurement.

ROI or delete all ROIs.

⑥ By checking ☑ Locked you can lock the selected ROI and thus prevent it from accidental editing.

⑦ ROIs can be selected by clicking next to it in the image. Alternatively, use the two arrow buttons ◀ ▶ to step through all the ROIs of the selected tool.

2.2 Extracting Edges Using and

While you created and modified ROIs, ActivMeasure already started to work and extracted edges along the ROIs, i.e., the lines of measurement. We now show how to specify what is to be extracted.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add `AVTMeasureGraph` by double-clicking . You may remove `AVTViewROI`.

  Otherwise, open the project `edges\measure_edges.vbp`.

- ☐ Execute the application (`Run` ▷ `Start` or via the corresponding button) and load the image `chip\chip_01`. If neccessary, place a line-shape ROI as shown in figure 2.2; `AVTViewFG` and `AVTViewROI` can be opened at run time via a right mouse button click on `AVTView`.

The following steps are visualized in figure 2.2.

① In the combo box `Edge Type` you can select the type of edges you want to be extracted, e.g., only edges where image intensity changes from light to dark. Note that the direction of measuring is indicated by an arrow on the ROI.

② In the combo box `Edge Grouping`, you can choose whether edges are to be grouped to pairs, i.e., one edge from light to dark and one from dark to light or vice versa. Note that if you choose this mode (`'Edge Pairs'`), the value selected in `Edge Type` determines whether grouping starts with the first edge (`'All Edges'`), with the first `'Light → Dark'`, or the first `'Dark → Light'` edge.

③ With the help of the combo box `Edge Selection`, you can further reduce the number of edges to the `'First Edge'`, the `'Last Edge'`, or both the `'First+Last Edge'`.

④ With the slider `Edge Strength` you can specify the minimum strength of the edges to be extracted. The strength of an edge corresponds to the gradient of intensity. This relation becomes clear when looking at the intensity profile depicted in `AVTMeasureGraph`: Edges are extracted at points where the intensity rises or falls quickly. Their strength is visualized by the bold bars; the red ones denote edges that are above the threshold, the magenta ones fall short of the threshold by up to ten percent.

Experiment with the slider to get a feeling for this parameter. You can also drag the horizontal line in `AVTMeasureGraph` to change the minimum strength. As a rule of thumb, keep the line a bit below the bars of those edges you want to be extracted and

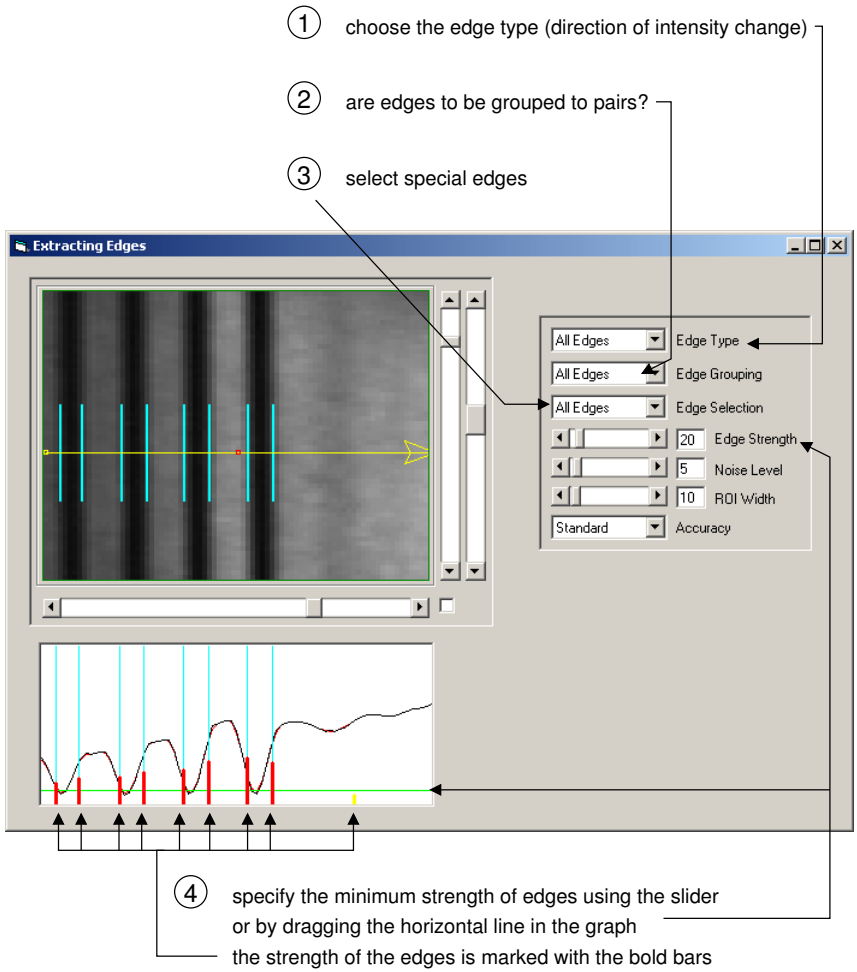


Figure 2.2: Specifying the edges that are to be extracted.

above those you do not want.

We continue with the example on the next double page.

(Extracting Edges, continued)

Visual Basic Example

Preparation for the following example:

- ☐ Place a line-shaped ROI over the radial conductor paths near the middle of the image. At first, not all edges are extracted.

The following steps are visualized in figure 2.3.

(5) If you have got a noisy image, it is useful to smooth it using the slider `Noise Level` before extracting edges. Graphically spoken, smoothing removes narrow peaks in the intensity profile. Experiment with this parameter; the smoothed profile is depicted in `AVTMeasureGraph` by a second profile drawn in red.

(6) If you chose a line-shaped or an arc-shaped ROI, you can specify the width of averaging with the slider `ROI Width`. For a rectangular ROI this slider is disabled and shows the width of the rectangle.

In our example, the averaging parameter prevents the radial edges from being extracted. Experiment with this parameter until all edges are successfully extracted.

Note that the ROI width can be set for each ROI individually; all other parameters can be set only once per tool.

(7) In the combo box labeled `Accuracy`, you can choose the method of averaging. In normal cases, `'Standard'` is sufficient; the other variants take considerably more computing time and should be used only in case of high accuracy measurements along arcs or in thin rectangles (or lines) that are oriented arbitrarily.

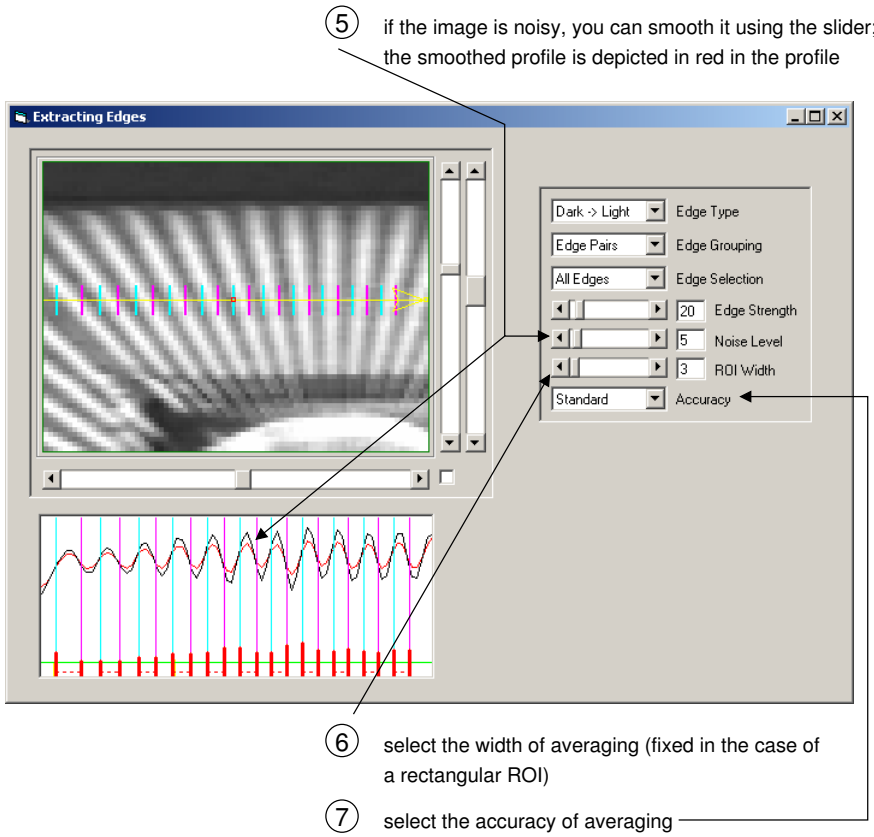


Figure 2.3: Specifying the edges that are to be extracted (cont.).

2.3 Selecting Results Using

Up to now we have just extracted edges, i.e., the starting points for our measurements. Using `AVTMeasureResults`, you can specify what is to be measured between the edges. These results (also called *features*) are then sent to subsequent ActivVisionTools, e.g., ActivDecision.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add `AVTMeasureResults` and ActivDataView to the form by double-clicking and , respectively. You may remove `AVTMeasureGraph`; it can be opened at run time by clicking on `AVTMeasure` with the right mouse button and selecting `Graph`. Execute the application (`Run` ▷ `Start` or via the corresponding button).
- ☐ Otherwise, open the project `results\measure_results.vbp`. Execute the application (`Run` ▷ `Start` or via the corresponding button).
- ☐ Load the image `chip\chip_01` and create one or more ROIs; `AVTViewFG` and `AVTViewROI` can be opened at run time via a right mouse button click on `AVTView`.

The following steps are visualized in figure 2.4.

① Select the features to be calculated by checking the corresponding box in `AVTMeasureResults`. Available features for single edges are their position in the image, their amplitude (= strength), and the distance to the next edge. If edges are grouped to pairs (i.e., if you selected `'Edge Pairs'` in the combo box `Edge Grouping`), additional features can be selected: the pair width, the position of the center of a pair, and the distance between pair centers. Furthermore, the angle of the edges and the angular distance between them can be calculated as well as the angular pair width, the angle of the pair centers, and the angular distance between successive pair centers. Note that the angular features provide meaningful information only for arcs of measurement; for lines, they are zero.

② To view the calculated features, enable the update of results in `AVTDataView` and select the tool whose results to display in the combo box `ActivVisionTool`. If there is only one tool, as in our example, it is selected automatically.

③ Automatically, a list of the ROIs of `AVTMeasure1` appears. Select the ROIs whose results you want to examine by checking their box. Their results are then displayed in a table (spreadsheet). The columns correspond to the selected features, the rows to the extracted *objects*. If you chose to group edges to pairs (i.e., if you selected `'Edge Pairs'` in the combo box `Edge Grouping`), each edge pair corresponds to an object; otherwise, objects correspond to single edges.

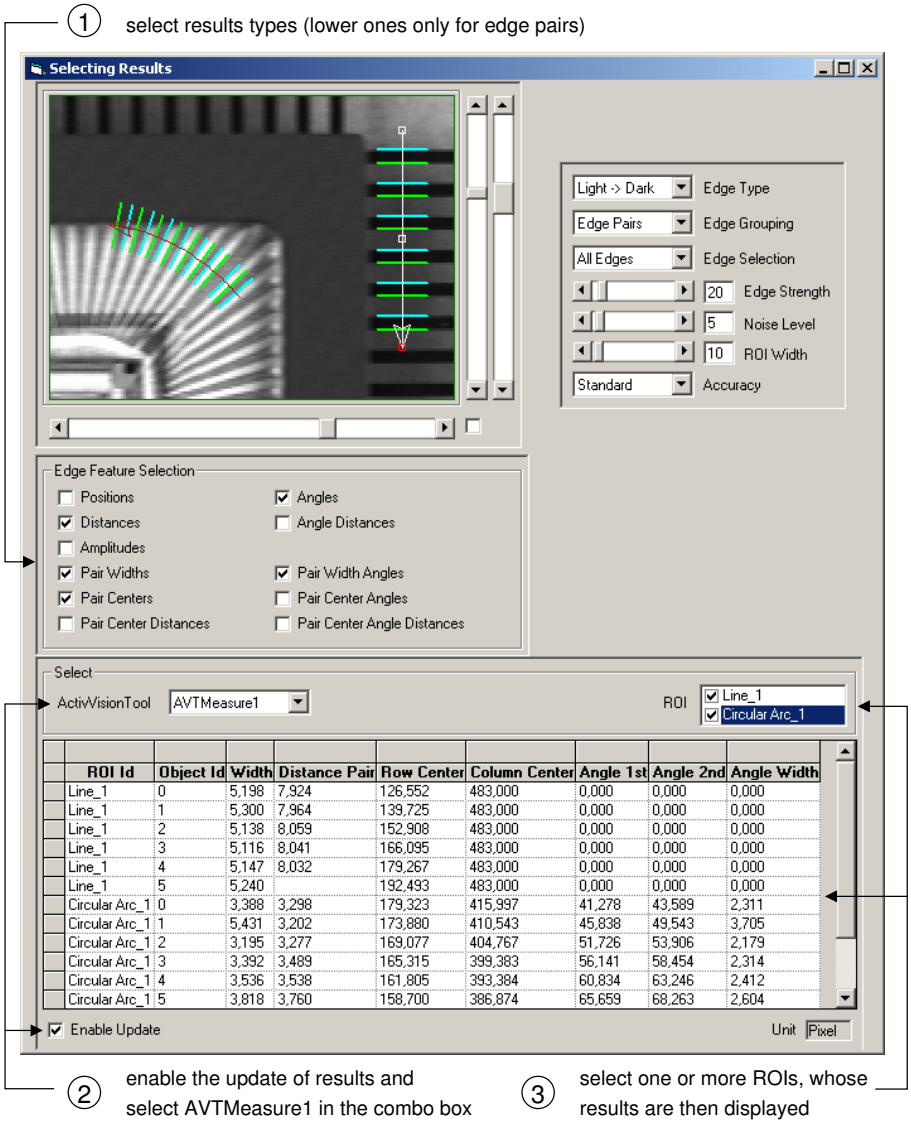

1 select results types (lower ones only for edge pairs)
Selecting Results
Light -> Dark
Edge Type
Edge Pairs
Edge Grouping
All Edges
Edge Selection
20
Edge Strength
5
Noise Level
10
ROI Width
Standard
Accuracy
Edge Feature Selection
Positions
Angles
Distances
Angle Distances
Amplitudes
Pair Widths
Pair Width Angles
Pair Centers
Pair Center Angles
Pair Center Distances
Pair Center Angle Distances
Select
ActivVisionTool
AVTMeasure1
ROI
Line_1
Circular Arc_1
ROI Id | Object Id | Width | Distance Pair | Row Center | Column Center | Angle 1st | Angle 2nd | Angle Width
Line_1 | 0 | 5,198 | 7,924 | 126,552 | 483,000 | 0,000 | 0,000 | 0,000
Line_1 | 1 | 5,300 | 7,964 | 139,725 | 483,000 | 0,000 | 0,000 | 0,000
Line_1 | 2 | 5,138 | 8,059 | 152,908 | 483,000 | 0,000 | 0,000 | 0,000
Line_1 | 3 | 5,116 | 8,041 | 166,095 | 483,000 | 0,000 | 0,000 | 0,000
Line_1 | 4 | 5,147 | 8,032 | 179,267 | 483,000 | 0,000 | 0,000 | 0,000
Line_1 | 5 | 5,240 | | 192,493 | 483,000 | 0,000 | 0,000 | 0,000
Circular Arc_1 | 0 | 3,388 | 3,298 | 179,323 | 415,997 | 41,278 | 43,589 | 2,311
Circular Arc_1 | 1 | 5,431 | 3,202 | 173,880 | 410,543 | 45,838 | 49,543 | 3,705
Circular Arc_1 | 2 | 3,195 | 3,277 | 169,077 | 404,767 | 51,726 | 53,906 | 2,179
Circular Arc_1 | 3 | 3,392 | 3,489 | 165,315 | 399,383 | 56,141 | 58,454 | 2,314
Circular Arc_1 | 4 | 3,536 | 3,538 | 161,805 | 393,384 | 60,834 | 63,246 | 2,412
Circular Arc_1 | 5 | 3,818 | 3,760 | 158,700 | 386,874 | 65,659 | 68,263 | 2,604
Enable Update
Unit
Pixel
2 enable the update of results and select AVTMeasure1 in the combo box
3 select one or more ROIs, whose results are then displayed

Figure 2.4: Selecting what is to be measured.

You can also display the distribution of the calculated feature values in a histogram using ActivFeatureHistogram, which can be opened by clicking on `AVTView` with the right mouse button and selecting `Feature Histogram` in the appearing context menu.

Chapter 3

Combining ActivMeasure with other ActivVisionTools

While the previous chapter explained how to get measurement results, this chapter focuses on how to further evaluate and output the results using other ActivVisionTools. How to access results and evaluations via the programming interface is described in chapter 4 on page 25.

In the corresponding Visual Basic projects, the task is to examine car fuses.

3.1 Converting Results to Other Units **18**
3.2 Evaluating Results **20**
3.3 Output of Results **22**

3.1 Converting Results to Other Units Using

In section 2.3 on page 14, distances and positions were measured in image coordinates, i.e., pixels. Using `AVTViewCalibration`, you can convert these measurements into other units. Note that for a more accurate calibration you should employ ActivGeoCalib.

The main idea behind `AVTViewCalibration` is that the user draws a line in the image and tells ActivVisionTools that the length of this line is in a certain unit. From this information, `AVTViewCalibration` calculates the size of a pixel (i.e., its height as square pixels are assumed) in this unit, which in its turn can be used to convert measurements from pixels into the new unit. Note that this conversion only works if the measured objects lie in the same plane, i.e., at the same distance from the camera, as the line whose length was specified.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the example in the previous chapter, you may continue using this project. At design time, add `AVTViewCalibration` to the form by double-clicking with the left mouse button. You may remove `AVTMeasureResults`; it can be opened at run time by clicking on `AVTMeasure` with the right mouse button and selecting `Features`.

  Otherwise, open the project `units\measure_units.vbp`.

- ☐ Execute the application (`Run` ▷ `Start` or via the corresponding button) and load the image `fuse\fuse25_01`. If necessary, place an ROI for `AVTMeasure` over the fuse wire as shown in figure 3.1; `AVTViewFG` and `AVTViewROI` can be opened at run time via a right mouse button click on `AVTView`.

The following steps are visualized in figure 3.1 (not shown: `AVTMeasure`).

1. You create the line of known length using `AVTViewROI`. First, select `AVTViewCalibration1` in the combo box `ActivVisionTool`.
2. To start creating the line, click .
3. To draw the line, move the mouse in the image while keeping the left mouse button pressed.
4. As before, you can position the line precisely by dragging the pick points. Place it as shown in the figure; this distance corresponds to `4.5 mm` on the car fuses.
5. Now, switch your attention to `AVTViewCalibration`. Select the desired unit (`mm`) in the combo box `Unit`. The labels below denoting the unit will automatically switch to the selected unit.

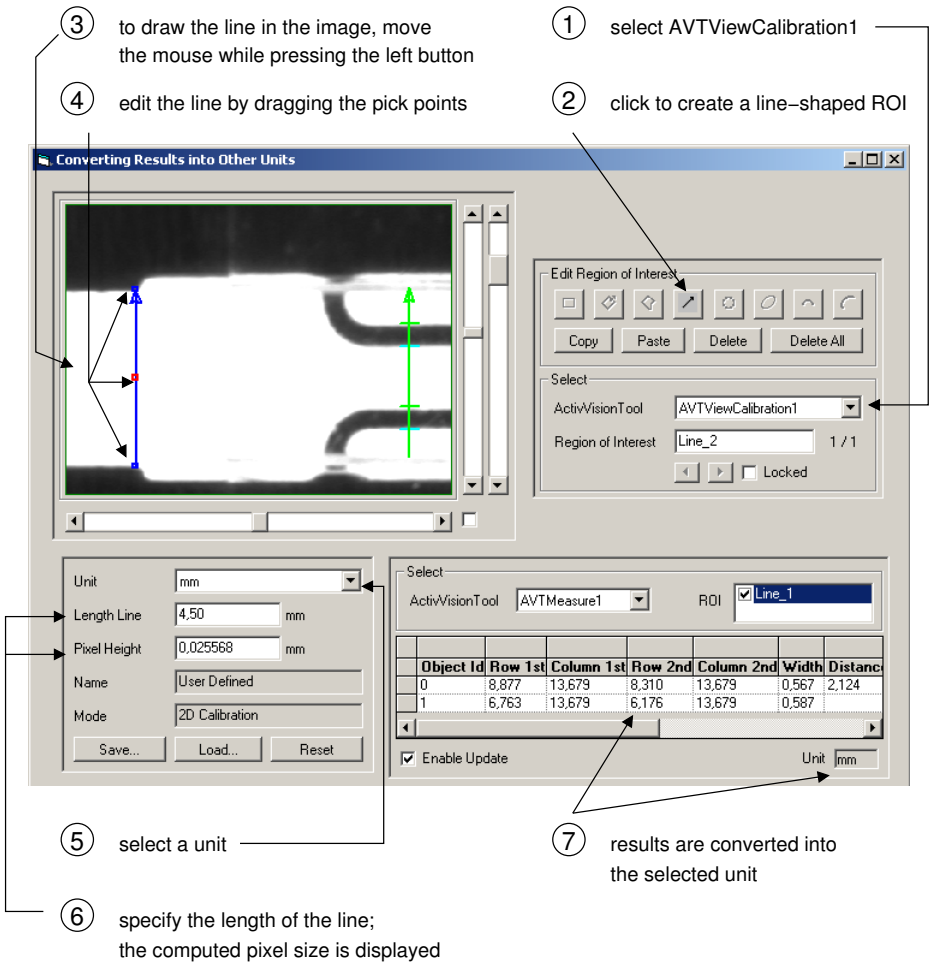


Figure 3.1: Converting measurements into other units.

(6) Then, specify the length of the line (`4.5`) in the text box `Length Line` and press `Enter`. `AVTViewCalibration` now calculates the height of a pixel in `mm` and displays it in the text box `Pixel Height`.

(7) Automatically, all measured distances and positions are converted into the selected unit.

3.2 Evaluating Results Using

In the former examples, you have employed ActivVisionTools to extract and display measurement results. Using ActivDecision, you can evaluate these results by formulating *conditions* the results have to meet in order to be "okay". For a detailed description of ActivDecision please consult the User's Manual for ActivDecision.

In our example measuring application, fuse types can be differentiated by the width of the wire; for 25 mA fuses, this width should lie between `0.5` and `0.6mm`.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add `AVTDecision` to the form by double-clicking with the left mouse button.

  Otherwise, open the project `decisions\measure_decisions.vbp`.

- ☐ Execute the application and load the image sequence `fuse\fuse1.seq`.

The following steps are visualized in figure 3.2 (not shown: `AVTView` and `AVTMeasure`).

① The main functionality of ActivDecision is presented by its *support tools*, which can be opened via clicking on `AVTDecision` with the right mouse button. The *master tool* displays the overall evaluation of the application.

② To view the current feature and evaluation results check ☑ `Enable Update` in `AVTDecisionViewResults`.

③ ActivDecision lets you compare the value of an individual object, ROI, or tool feature with two boundary values, a minimum and a maximum value. For our example application, change the minimum value for the feature `Width` to `0.5` and the maximum value to `0.6` in the columns `Min` and `Max`. Now, select a condition that the result has to meet in the column `Operation`; if you select `None`, the feature is not evaluated. In our example, the `Width` has to be `Inside` the two boundary values. Furthermore, exactly two objects have to be extracted, and none should be evaluated as "not okay".

④ Those features which meet their condition appear in green, the others in red. If at least one feature is "not okay", the whole object, ROI, or tool is evaluated as "not okay" as well. Analogously, the overall evaluation of application, which is visualized by `AVTDecision`, depends on the tool evaluations.

Figure 3.2 shows suitable conditions for the example task. When you step through the image sequence by clicking `Single` in `AVTViewFG`, you see that some fuses do not

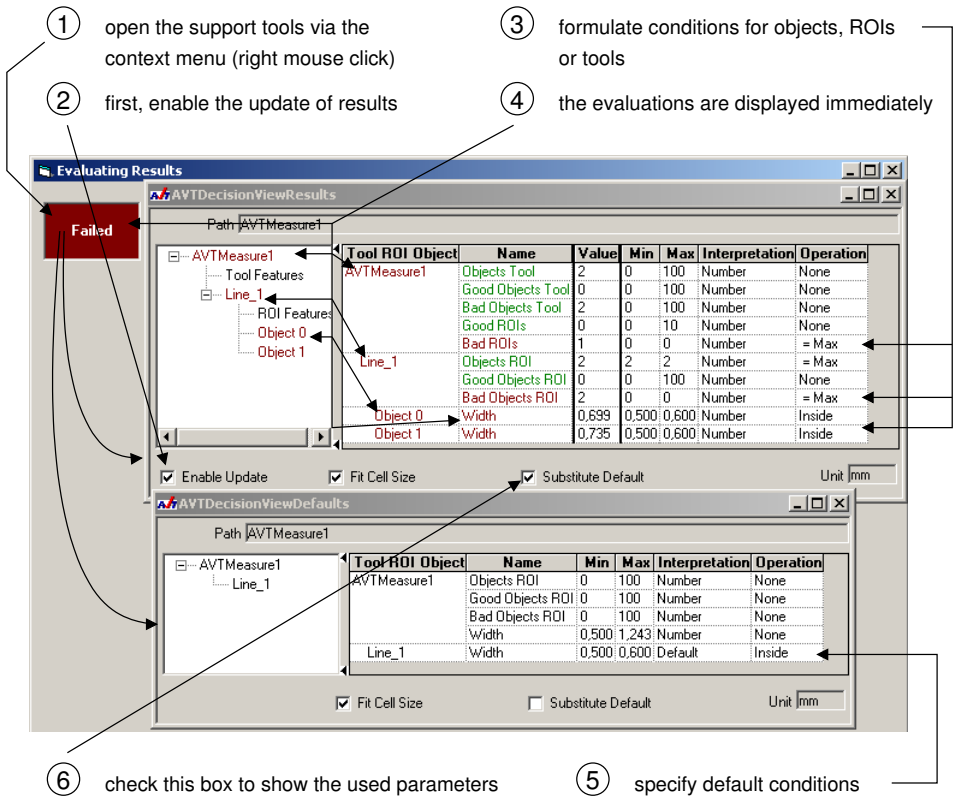


Figure 3.2: Formulating conditions to evaluate results.

meet the conditions.

(5) In the example application, both extracted wires should meet the same condition. Instead of specifying the same conditions for each object, you can specify default conditions using `AVTDecisionViewDefaults`. Defaults can be set per tool or per ROI; ROI defaults override tool defaults, and individual conditions override defaults.

(6) If you check ☑ `Substitute Default`, the entries marked `Default` are substituted by their actual content.

3.3 Output of Results Using

Using ActivFile, you can write the results and the evaluations to a log file. How to access results via the programming interface is described in section 4.3 on page 30, how to output them via a serial interface or a digital I/O board in the User's Manual for ActivSerial and the User's Manual for ActivDigitalIO, respectively.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add `AVTOutputFile` by double-clicking .

  Otherwise, open the project `output\measure_output.vbp`.
- ☐ Execute the application and load the image sequence `fuse\fuse1.seq`.

The following steps are visualized in figure 3.3 (not shown: `AVTView`, `AVTMeasure`).

① By clicking on `Select`, you can open a file selector box to choose a file name for the log file, which will then appear in the text field beside the button. By pressing `Clear File`, you can clear the content of the selected file.

② By checking ☑ `Enable Writing` you enable the writing mode.

③ You can open the ActivFile's two dialogs `DialogFileOptions` and `DialogOutputDataSelect` by clicking `File Options` and `Data Selection`, respectively.

④ In `DialogFileOptions`, you can choose between two file formats: Standard text (suffix `.txt`) and the so-called *comma-separated values* files (suffix `.cvs`) which can be used as an input to Microsoft Excel. Furthermore, you can select a delimiter.

⑤ In the same dialog you can limit the size of the log file in form of the number of *cycles* that are to be recorded. A cycle corresponds to one processing cycle from image input to the evaluation and output of results. If you use this option, ActivFile creates two log files and switches between them, thus assuring that you can always access (at least) the results of the last N cycles, N being the specified number of cycles.

⑥ By pressing `Estimate`, you can let ActivFile estimate the size of one cycle. Note that you must first select the output data in order to get meaningful results!

⑦ In the left part of `DialogOutputDataSelect`, you can navigate through the result hierarchy similarly to ActivDecision.

⑧ In the right part of `DialogOutputDataSelect`, select the output data by checking the corresponding boxes. You may output different items depending on the evaluation of an

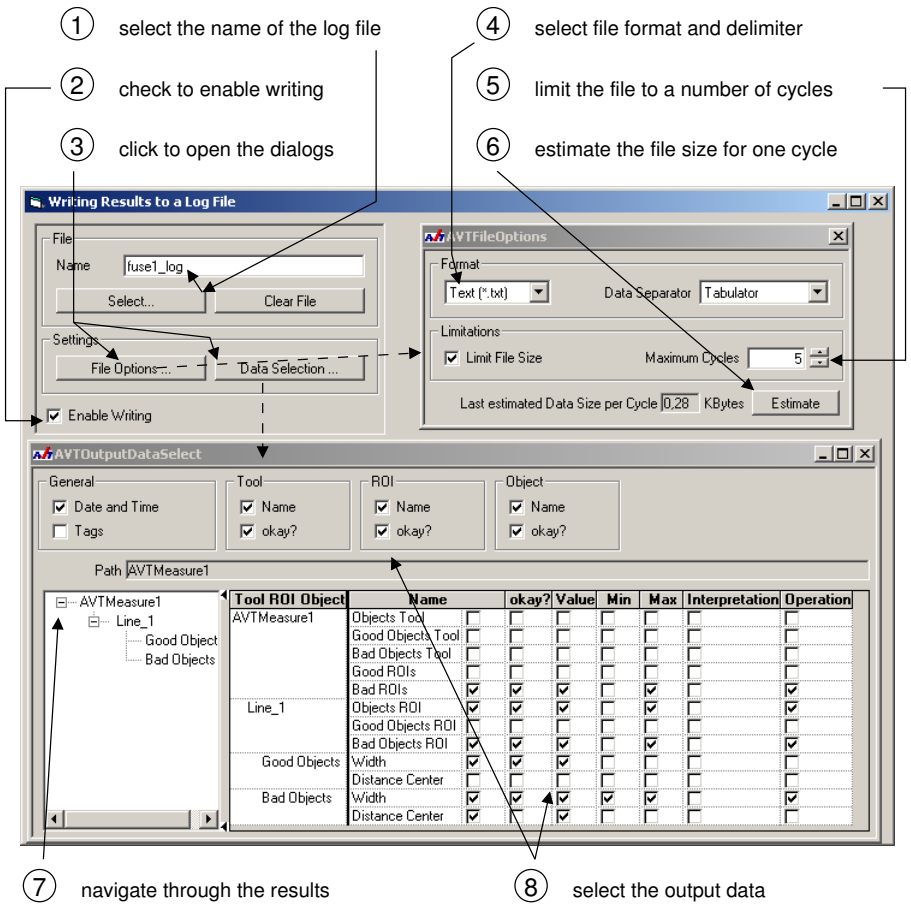


Figure 3.3: Customizing log files.

object. By clicking on the column labels with the right mouse button you can check or uncheck all boxes in the column; similarly, you can check or uncheck whole rows or all rows of a certain tool.

If you now step through the image sequence by clicking `Single` in `AVTViewFG`, the log file is created. Figure 3.4 shows part of an example log file.

```
09/11/04  15:26:10
AVTMeasure1  yes
      Bad ROIs          yes   0  0   = Max
  Line_1  yes
      Objects ROI       yes   2  2   = Max
      Bad Objects ROI   yes   0  0   = Max
    0  yes
      Width             yes   0.551424
    1  yes
      Width             yes   0.571242

09/11/04  15:26:11
AVTMeasure1  yes
      Bad ROIs          yes   0  0   = Max
  Line_1  yes
      Objects ROI       yes   2  2   = Max
      Bad Objects ROI   yes   0  0   = Max
    0  yes
      Width             yes   0.536949
    1  yes
      Width             yes   0.555279

09/11/04  15:26:11
AVTMeasure1  no
      Bad ROIs          no  1  0   = Max
  Line_1  no
      Objects ROI       yes   2  2   = Max
      Bad Objects ROI   no  2  0   = Max
    0  no
      Width             no  0.698542  0.500000  0.600000  Inside
      Distance Center   2.223346
    1  no
      Width             no  0.734562  0.500000  0.600000  Inside
```

Figure 3.4: Part of an example log file.

Chapter 4

Tips & Tricks

This chapter contains additional information that facilitates working with ActivMeasure, e.g., how to modify the graphical display of results and to customize the appearance of an ActivMeasure application in the two execution modes. Furthermore, it shows how to access measurement and evaluation results directly via the programming interface of ActivVisionTools.

4.1 Adapting the Display of Results 26
4.2 Configuring the Two Execution Modes 28
4.3 Accessing Results Via the Programming Interface 30
4.3.1 Measurement Results 31
4.3.2 Evaluation Results 36

4.1 Adapting the Display of Results Using

You can adapt the way results and ROIs are displayed using `AVTViewDisplayModes`, which is a *support tool* of ActivView.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the example in the previous chapter, you may continue using this project. Otherwise, open the project `display\measure_display.vbp` and execute it (`Run` ▷ `Start` or via the corresponding button).
- ☐ Load the image sequence `fuse\fuse1.seq` and create additional ROIs; `AVTViewFG` and `AVTViewROI` can be opened at run time via a right mouse button click on `AVTView`.

The following steps are visualized in figure 4.1. Experiment by choosing different settings for the display parameters and watching the result.

1. Open `AVTViewDisplayModes` by clicking on `AVTView` with the right mouse button and selecting `Display Modes` in the popup menu.
2. Change the color of the selected pick point. Select another pick point by clicking into its vicinity.
3. Change the color of the currently selected ROI. Select another ROI by clicking into its vicinity. Change the color of the other, not selected ROIs of the currently selected tool. Change the color of the ROIs of the other, not selected tools. As a second tool, you can open `AVTViewCalibration` via a right mouse button click on `AVTView` and create an ROI for it. If you use ActivDecision to evaluate results, it can mark ROIs evaluated as "not okay" in a special color.
4. Change the LUT (look-up table) that is used to display the image.
5. Change the line width or the ROIs or of the edges.
6. ActivVisionTools assigns results to two groups to allow to display them using different colors. If you selected `'All Edges'` in the combo box `Edge Grouping` of `AVTMeasure`, all extracted edges belong to group I. In contrast, if you selected `'Edge Pairs'`, the first edge of each pair is assigned to group I and the second edge to group II. Change the color of these groups.
7. Change the color ActivDecision uses to mark objects evaluated as "not okay".
8. Change the color used for highlighting objects if you click on them in ActivDataView or

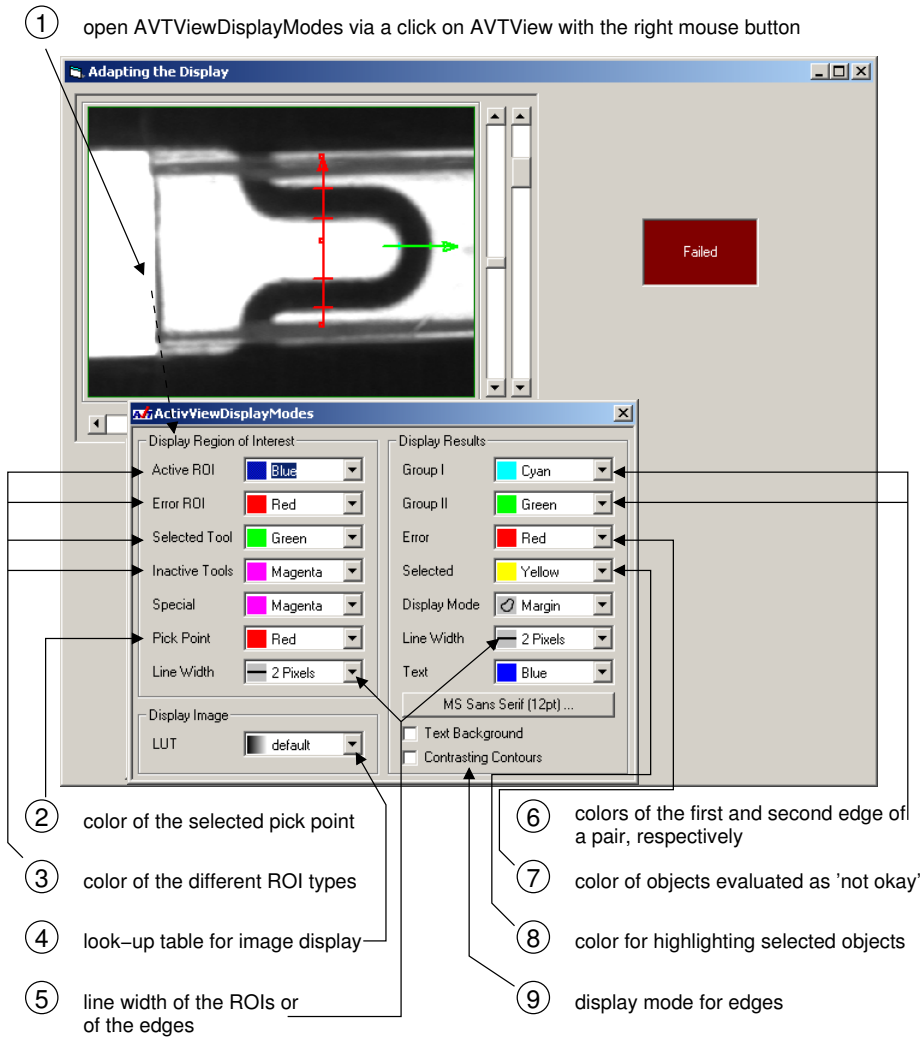


Figure 4.1: Modifying the display of results and ROIs .

in the tree of ActivDecision. To test the effect, open ActivDataView by clicking on `AVTMeasure` with the right mouse button and selecting `Data View` in the appearing context menu; then, click on one of the objects, which is automatically highlighted in `AVTView`.

(9) If you check this box, the edges are framed with a contrasting color.

4.2 Configuring the Two Execution Modes Via and

In an ActivVisionTools application you can switch between two execution modes: the *configuration mode* and the *application mode*. The former should be used to setup and configure an application, the latter to run it. ActivView's *support tools* `AVTViewExecute` and `AVTViewConfigExec` allow you to switch between the two modes and to customize the behavior of an ActivVisionTools application in the two execution modes, e.g., display live images only in the *configuration mode* to setup your application, but then switch it off in the *application mode* to speed up the application. A third sub-tool, `AVTViewExecuteSimple`, provides a single button to execute the application. `AVTViewStatus`, another sub-tool of ActivView, lets you inspect the status of your application.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add `AVTViewExecuteSimple` and `AVTViewStatus` to the form by double-clicking the icons and with the left mouse button.

  Otherwise, open the project `usermodes\measure_usermodes.vbp`.

- ☐ Execute the application and load the image sequence `fuse\fuse1.seq`.

The following steps are visualized in figure 4.2 (not shown: `AVTMeasure`).

1. Open `AVTViewExecute` and `AVTViewConfigExec` by clicking on `AVTView` with the right mouse button and selecting `Execution` and `Execution Parameters`.
2. Switch between the two execution modes via `AVTViewExecute`'s combo box `Mode`.
3. To execute one cycle, press `Single`. With the other two buttons you can let the application run continuously and stop it again.

   By default, `AVTViewExecuteSimple` starts and stops an application; how to change its behavior to a single-step button is described in the User's Manual for ActivView, section 3.4 on page 34.
4. For each of the two execution modes, you can choose what is to be displayed by checking the corresponding boxes in `AVTViewConfigExec`. Furthermore, you can specify if images can be dragged to the image window and whether ROIs can be modified in the two modes; by default, this is disabled in the *application mode* to prevent you from accidentially moving or deleting an ROI.
5. In `AVTViewStatus`, an icon indicates the current execution mode of the application.

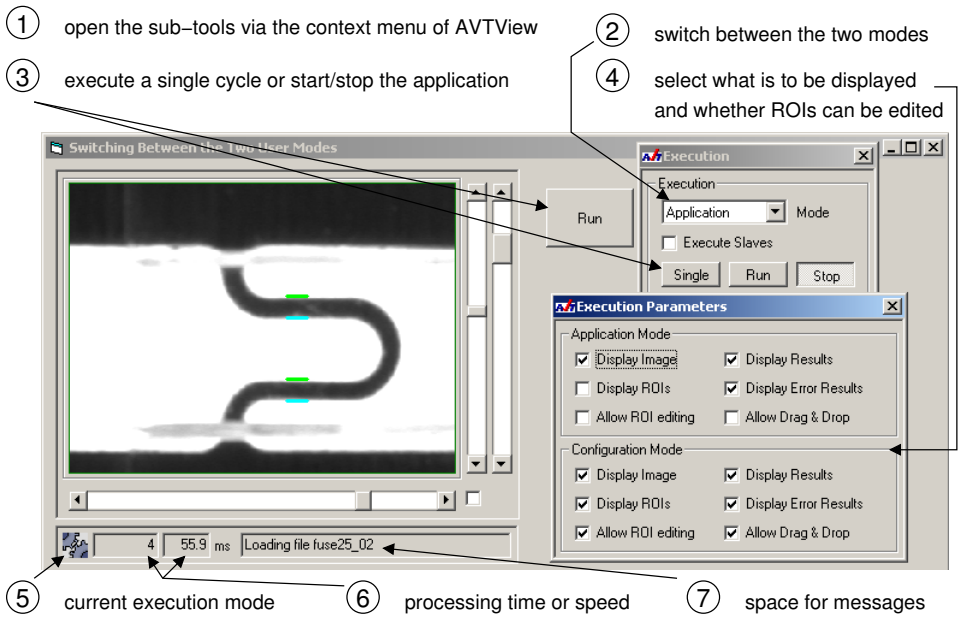


Figure 4.2: Customizing and switching between the two execution modes.

In the mode , the application does not perform any processing and waits for your interaction. If you start the continuous mode the cogwheels rotate; any interaction on your part is stored in the event queue and processed after the current cycle is finished. If the cursor gets “busy”, ActivVisionTools has started a particularly time-consuming operation, e.g., connecting to an image acquisition device. Any interaction on your part is then deferred to the end of this operation.

(6) `AVTViewStatus` also shows the number of processed cycles and the time needed for the last processing cycle.

(7) `AVTViewStatus` display two types of messages: Informative messages describe, e.g., what the application is doing while it is “busy”, while error messages indicate errors that prevent the application from working correctly, e.g., the failure to connect to an image acquisition device.

If `AVTViewStatus` is not added to an application, error messages are displayed in popup dialogs.

More information about `AVTViewStatus`, e.g., how to modify its appearance, can be found in the User’s Manual for ActivView, section 3.3 on page 32.

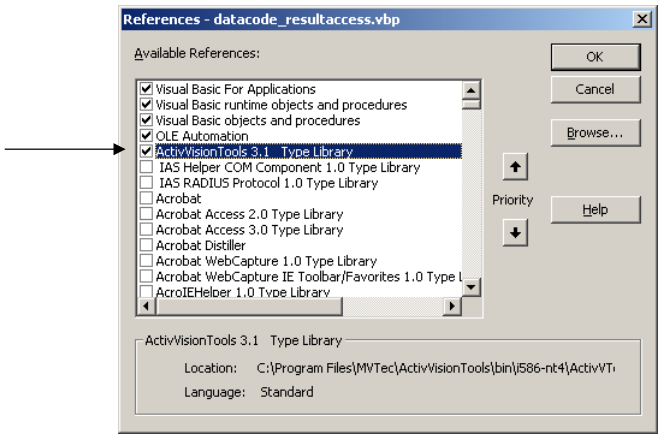
4.3 Accessing Results Via the Programming Interface

The previous chapters and sections showed how to use ActivVisionTools interactively, i.e., via the graphical user interfaces presented by the underlying ActiveX controls. In this mode, you can develop the image processing part of your machine vision application rapidly and easily, without any programming. However, there is more to ActivVisionTools than the graphical user interfaces: Because ActivVisionTools comes as a set of ActiveX controls, it provides you with an open programming interface, thereby offering full flexibility.

In this section, we show how to access the measurement and evaluation results via the programming interface. With this, you can, e.g., realize an application-specific graphical user interface, perform additional processing on the results, or send results to a special output device. Detailed information about the programming interface can be found in the **Reference Manual**.

As in the previous sections, the examples stem from Visual Basic 6.0; if the (ActivVisionTools-specific) code differs in Visual Basic .NET, the corresponding lines are also shown (for the first appearance only). For other .NET languages or C++, please refer to the Advanced User's Guide for ActivVisionTools, section 1.2.3 on page 5 and section 1.3.4 on page 28, respectively. Please note that we assume that readers of this part have at least a basic knowledge of Visual Basic.

To work with the programming interface, in VB 6.0 you must first **add the ActivVisionTools type library to the project's references** by checking the box labeled `ActivVisionTools Type Library` in the menu dialog `Project` ▷ `References`. In Visual Basic .NET, the reference is added automatically.



4.3.1 Accessing Measurement Results

The principal idea behind accessing the results of an ActivVisionTool is quite simple: When a tool has finished its execution, it raises an event called `Finish`, sending its results as a parameter. If you want to access the results, all you have to do, therefore, is to create a corresponding event procedure which handles the event.

Within the Visual Basic environment, you can create event procedures very easily as shown in figure 4.3: In the header of the form's code window there are two combo boxes. Select the instance of `AVTMeasure` (by default called `AVTMeasure1`) in the left combo box. The right combo box then lists all events available for this object; when you select `Finish`, the event procedure is created automatically. Within this procedure, the measurement results are now accessible via the object variable `atToolResults`.

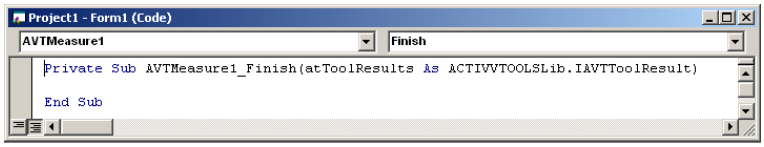


Figure 4.3: Creating a procedure to handle the event `Finish` .

`atToolResults` contains the result data for all ROIs of your instance of `AVTMeasure`. The current number of ROIs can be queried via

```
Dim iNumROIs As Integer

iNumROIs = atToolResults.ROINum
```

In Visual Basic .NET , the event handler has a different signature:

```
Private Sub AxAVTMeasure1_Finish(ByVal sender As Object, _
                                 ByVal e As _
                                   AxActivVTools.__AVTMeasure_FinishEvent) _
                                 Handles AxAVTMeasure1.Finish
```

A first difference is that tool names start the prefix `Ax`, i.e., `AVTMeasure` becomes `AxAVTMeasure`. The main difference, however, is that the tool results are not directly passed; instead, they are encapsulated in the parameter `e`. From there, they can be extracted with the following lines:

```
Dim atToolResults As AVTToolResult
atToolResults = e.atToolResults
```

To use classes like `ACTIVVTOOLSLib.AVTToolResult` without the namespace `ACTIVVTOOLSLib` as in the code above, you must import this namespace by inserting the following line at

the very beginning of the code (more information about importing namespaces can be found in the Advanced User's Guide for ActivVisionTools in section 1.2.4.5 on page 12):

```
Imports ACTIVVTOOLSLib
```

The results of a certain ROI can be accessed by specifying its name in a call to the method `ROIResult`, or by specifying its index in a call to the method `ROIResults`. The following code uses the latter method to access the first ROI of `AVTMeasure1`:

```
Dim atROIResult As AVTROIResult

Set atROIResult = atToolResults.ROIResults(0)
```

Now, we can, e.g., query the number of objects extracted in the ROI via

```
Dim iNumObjects As Integer

iNumObjects = atROIResult.ObjectNum
```

Actual measurement results for an object, i.e., the calculated values of features like `Distance` or `Width` can be accessed by specifying the feature of interest and the ID of the object in a call to the method `ObjectValue` of `ACTIVVTOOLSLib.AVTROIResult`. The feature handles are available as methods of the corresponding tool, e.g., `AVTMeasure.FeatureHandlePairWidth` being the handle for the measured width.

The following code fragment uses another method of `ACTIVVTOOLSLib.AVTROIResult`, `ObjectValues`, which returns the values of all objects for the specified feature in an array, to calculate the mean width:

```
Dim handleWidth As Integer, i As Integer
Dim vWidthArray As Variant
Dim dSumWidth As Double, dMeanWidth As Double

handleWidth = AVTMeasure1.FeatureHandlePairWidth
vWidthArray = atROIResult.ObjectValues(handleWidth)
dSumWidth = 0
For i = 0 To iNumObjects - 1
  dSumWidth = dSumWidth + vWidthArray(i)
Next
dMeanWidth = dSumWidth / iNumObjects
```

A general difference **in Visual Basic .NET** is that instead of the type `Variant` you must use `Object` when accessing multiple values:

```
Dim vWidthArray As Object
```

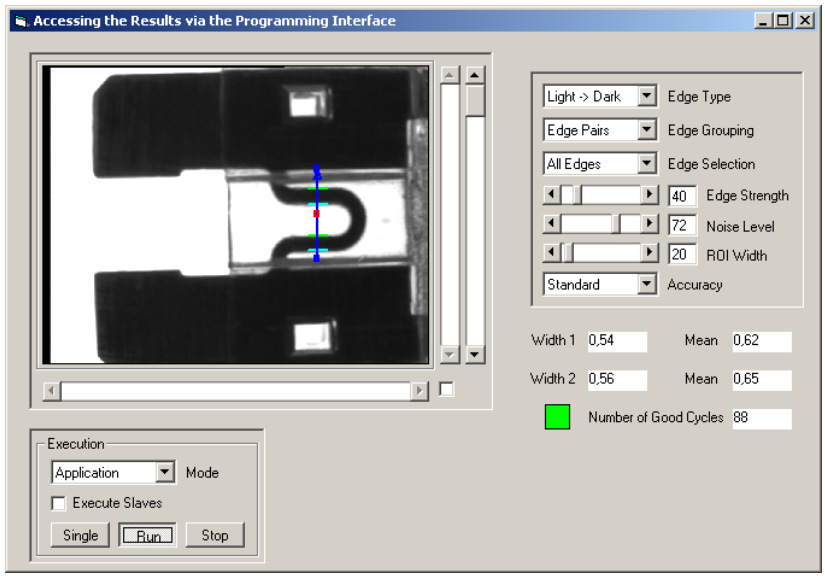


Figure 4.4: Accessing and displaying the measurement results.

The ActivVisionTools distribution includes the example Visual Basic project `resultaccess\measure_resultaccess.vbp`, which uses the methods described above to extend the application for inspecting car fuses introduced in the previous chapter. The task is to display the measured widths of the wire and their mean values over time (see figure 4.4). The example project is already configured, just start it and click the button `Run` in `AVTViewExecute`.

Besides accessing the measurement results, the project code contains additional functionality which is explained briefly in the following. Note that the code is only shown for Visual Basic 6.0; a Visual Basic .NET application with result access can be found in the directory `examples\dotnet\vb\measure_results`.

First of all, the measurement results are only to be read and displayed if exactly two objects have been extracted:

```
If iNumObjects = 2 Then
  dWidth1 = atROIResult.ObjectValue(handleWidth, 0)
  dWidth2 = atROIResult.ObjectValue(handleWidth, 1)
  TextWidth1.Caption = Format(dWidth1, "Fixed")
  TextWidth2.Caption = Format(dWidth2, "Fixed")

  If bIsError = True Then
    Call ClearAlarm
  End If

Else
  TextWidth1.Caption = "---"
  TextWidth2.Caption = "---"
  Call SetAlarm
End If
```

If not exactly two objects are extracted, the function `SetAlarm` stops the application by setting `AVTView`'s property `RunState` to `'False'` and switches the color of the element beside the number of good cycles to red. The function `ClearAlarm` resets the color to green. Test this behavior by modifying the ROI such that in some images only one object is extracted and then starting the continuous execution.

```
Private bIsError As Boolean

Private Function SetAlarm()
  AVTView1.RunState = False
  Light.BackColor = vbRed

Private Function ClearAlarm()
  Light.BackColor = vbGreen
  bIsError = False
End Function
```

When calculating the mean values, one has to keep in mind that `AVTMeasure` is executed not only when the next image is grabbed but also whenever you modify its ROI (s) or parameters. To distinguish the two cases an event raised by `AVTView` at the start of each execution cycle can be used to set a variable called `bIsNewCycle`:

```
Private bIsNewCycle As Boolean

Private Sub AVTView1_CycleStart()
  bIsNewCycle = True
End Sub
```

Before calculating the mean values within the handler for `AVTMeasure`'s event `Finish`, this

variable is checked (and immediately reset).

```
If bIsNewCycle = True Then
  iNumGoodCycles = iNumGoodCycles + 1
  bIsNewCycle = False
  dSumWidth1 = dSumWidth1 + dWidth1
  dMeanWidth1 = dSumWidth1 / iNumGoodCycles
  dSumWidth2 = dSumWidth2 + dWidth2
  dMeanWidth2 = dSumWidth2 / iNumGoodCycles
  TextMeanWidth1.Caption = Format(dMeanWidth1, "Fixed")
  TextMeanWidth2.Caption = Format(dMeanWidth2, "Fixed")
End If
```

You can test this behavior by moving or modifying the ROI: The new widths are displayed, while the mean values remain constant.

When using the programming interface of ActivVisionTools, you leave the safe world of the graphical user interfaces where all input is checked for validity automatically. In contrast, if you try to access a non-existent object or result via the programming interface, a run-time error is caused which terminates your application! To avoid this, you can use the Visual Basic error handling mechanisms, i.e., set up an error handler which examines any occurring error and reacts in a suitable way. In the example project, if an error is caused by the result access, a dialog with the error description pops up and the function `SetAlarm` is called. To view the effect of the error handler, de-select the feature `Width` in `AVTMeasureResults`.

```
Private Sub AVTMeasure1_Finish(atToolResults As _
                               ACTIVVTOOLSLib.IAVTToolResult)
  ' variable declarations

On Error GoTo ErrorHandler
  ' procedure body
  Exit Sub

ErrorHandler:
  Dim sTitle As String
  If Left(Err.Source, 11) = "ActivVTools" Then
    sTitle = "ActivVisionTools Error"
  Else
    sTitle = "Runtime Error " & CStr(Err.Number)
  End If
  Call MsgBox(Err.Description, vbExclamation, sTitle)
  Call SetAlarm
End Sub
```

By placing the following code at the beginning of the event handler, the actual result access is restricted to the *application mode*. With this mechanism you can setup the vision part of your

application in the configuration mode without having to worry about run-time errors.

```
If Not AVTView1.ExecutionMode = eApplicationMode Then
  TextWidth1.Caption = "---"
  TextWidth2.Caption = "---"
  Exit Sub
End If
```

4.3.2 Accessing Evaluation Results

The evaluation results can be accessed similarly to the measurement results; in fact, they are even stored in the same object. However, to access the evaluation results you now have to wait for ActivDecision to finish, i.e., create the following event procedure:

```
Private Sub AVTDecision1_Finish(atToolResults As Collection)

End Sub
```

Note that **you will get a run-time error if you try to access evaluation results before ActivDecision has finished** (e.g., in the handler for `AVTMeasure`'s event `Finish`)!

Because ActivDecision can evaluate the results of more than one tool, the event handler provides you with a `Collection` of tool results. The following code fragment searches the collection for the results of `AVTMeasure1` and "stores" them in the object variable `atMeasureResult`, or exits if no results are found:

```
Dim atToolResult As AVTToolResult
Dim atMeasureResult As AVTToolResult
Dim bMeasureResultsFound As Boolean

bMeasureResultsFound = False
For Each atToolResult In atToolResults
  If atToolResult.Name = "AVTMeasure1" Then
    Set atMeasureResult = atToolResult
    bMeasureResultsFound = True
  End If
Next
If bMeasureResultsFound = False Then
  Exit Sub
End If
```

In Visual Basic .NET , the event procedure has the following signature:

```
Private Sub AxAVTDecision1_Finish(ByVal sender As System.Object, _
                                  ByVal e As _
                                    AxActivVTools.__AVTDecision_FinishEvent) _
                                  Handles AxAVTDecision1.Finish
```

Again, the tool results are encapsulated in the parameter `e`. They can be extracted as follows; note the use of `VBA.Collection` instead of `Collection`!

```
Dim atToolResults As VBA.Collection
atToolResults = e.atToolResults
```

As already remarked in the previous section, tool names are prefixed with `Ax`, thus you must search for the results of `AxAVTMeasure1`:

```
If atToolResult.name = "AxAVTMeasure1" Then
```

You can query the overall evaluations at different levels, tool, ROI, or object:

```
Dim atROIResult As AVTROIResult
Dim bToolIsOK As Boolean, bROIIsOK As Boolean, bObjIsOK As Boolean

Set atROIResult = atMeasureResult.ROIResults(0)

bToolIsOK = atMeasureResult.Evaluation
bROIIsOK = atROIResult.Evaluation
bObjIsOK = atROIResult.ObjEvaluation(0)
```

Furthermore, you can access the evaluation of individual features like the measured width of the object with the ID `0`, but also of tool features (e.g., the evaluation of the number of “bad” ROIs) or of ROI features (e.g., the evaluation of the number of objects in the ROI) via the corresponding feature handle. In contrast to object features, the handles for tool and ROI features are available as properties of `ACTIVVTOOLSLib.AVTToolResult`.

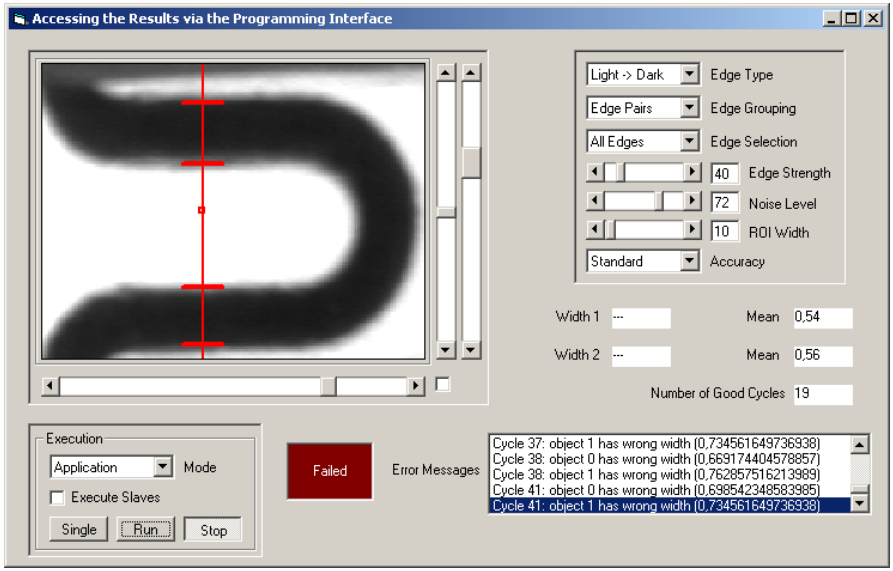


Figure 4.5: Accessing and displaying the evaluation results.

```
Dim handleWidth As Integer, handleBadROIs As Integer
Dim handleObjectsInROI As Integer
Dim bWidthIsOK As Boolean, bBadROIsIsOK As Boolean
Dim bObjectsInROIIsOK As Boolean

handleWidth = AVTMeasure1.FeatureHandlePairWidth
handleBadROIs = atMeasureResult.FeatureHandleBadROIs
handleObjectsInROI = atMeasureResult.FeatureHandleObjectsROI

bWidthIsOK = atROIResult.ObjFeatureEvaluation(handleWidth, 0)
bBadROIsIsOK = atMeasureResult.FeatureEvaluation(handleBadROIs)
bObjectsInROIIsOK = atROIResult.FeatureEvaluation(handleObjectsInROI)
```

The ActivVisionTools distribution includes the example Visual Basic project `evalaccess\measure_evalaccess.vbp`, which uses these methods to extend the application described in the previous section. Now, ActivDecision is used to check the width of the fuse wire as described in section 3.2 on page 20; in case the overall evaluation shows an error, the cause of the error is to be displayed (see figure 4.5). The example project is already configured, just start it and click the button `Run` in `AVTViewExecute`.

Besides accessing the evaluation results, the project code contains additional functionality which is explained briefly in the following (again only for Visual Basic 6.0!). First of all, the main `If -`

`Then - Else` clause around the display of measurement results now tests the overall evaluation of `AVTMeasure1`:

```
If atMeasureResult.Evaluation = True Then
  ' access and display measurement results
Else
  ' examine cause of error more closely
End If
```

In the example application, the only cause for `AVTMeasure1` being evaluated as "not okay" is that its (one) ROI has been evaluated as "not okay". This in turn can have two possible reasons: First, that the wrong number of objects has been extracted (i.e., less or more than two). In this case, the following code fragment displays a corresponding message in a list box which indicates how many objects have been extracted:

```
If atROIResult.FeatureEvaluation(handleObjectsInROI) = False Then
  errorMsg = "Cycle " & iNumCycles & ": wrong number of objects (" & _
             atROIResult.Value(handleObjectsInROI) & ")"
  Call DisplayMessage(errorMsg)
End If
```

The error message also contains the number of the cycle in which the error occurred. The corresponding counter is incremented in the handler for `AVTView`'s event `CycleStart` which was introduced already in the previous section:

```
Private iNumCycles As Integer

Private Sub AVTView1_CycleStart()
  iNumCycles = iNumCycles + 1
End Sub
```

A second cause of error is that at least one object has been evaluated as "not okay". Again, a corresponding error message is created, this time containing the measured width of the object. The following code shows how to access the indices of all objects evaluated as "not okay":

```
Dim badObject As Variant

If atROIResult.FeatureEvaluation(handleBadObjectsInROI) = False Then
  ' get all bad objects (cause: bad width)
  For Each badObject In atROIResult.BadObjectIndices
    errorMsg = "Cycle " & iNumCycles & ": object " & badObject & _
               " has wrong width (" & _
               atROIResult.ObjectValue(handleWidth, badObject) & ")"
    Call DisplayMessage(errorMsg)
  Next
End If
```